UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/692,200	10/23/2003	Sujal S. Parikh	14917.0230US01/MS305926.0	8417

27488 7590 05/28/2009
MERCHANT & GOULD (MICROSOFT)
P.O. BOX 2903
MINNEAPOLIS, MN 55402-0903

EXAMINER	
AUGUSTINE, NICHOLAS	

ART UNIT	PAPER NUMBER
2179	

MAIL DATE	DELIVERY MODE
05/28/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/692,200	PARIKH ET AL.	
	Examiner	Art Unit	
	NICHOLAS AUGUSTINE	2179	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on _10 March 2009_.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) _1-4,7-18,20-24,26-29,32 and 33_ is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) _1-4, 7-18, 20-24, 26-29 and 32-33_ is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _3/10/2009_.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

A. This action is in response to the following communications: Request for Continued Examination filed 3/10/2009.

B. Claims 1-4, 7-18, 20-24, 26-29 and 32-33 remains pending.

Continued Examination Under 37 CFR 1.114

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 3/10/2009 has been entered.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

4. Claims 1-4, 7-10, 12-18, 20-21, 23-24, 26-29, 32-33 are rejected under 35 U.S.C. 103(a) as being unpatentable over Breinberg (US Pat. 5,886,694) in view of Buxton et al (US Pat. 6,469,714).

As claim 1, Breinberg teaches a method of making ready for presentation a graphical element in a computer application program by communicating with a computer operating system (col. 1, lines 59-62; col. 13, lines 34-39), the method comprising: maintaining a measure tree storing a list of elements to be measured (col.4, lines 31-45 (specification stage; during the specification stage graphical user interface elements are specified according to parent and child relationships and their boundaries, also the size of the elements are specified (col.6,lines 65-67; col.7,lines 1-16). The data

structured used to access these elements is a tree data structure)); maintaining an arrange tree storing a list of elements to be arrange (col.4, lines 46-56 (layout stage; during the layout stage the method access a tree data structure to go through all of the graphical user interface elements using an autolayout engine that traverses the tree data structure; from here the graphical user interface is constructed with appropriate layout of elements (col.9, lines 37-47)); executing a first procedure for measuring the element, wherein the first procedure at least determines whether the element has one or more children ,determines, if the element has one or more children, whether the one or more children of the element is to be measured (col.4, lines 56-64); and determines a size for the element based on an element type for the element when the element has no children (fig. 6, label 602; col. 2, lines 1-9; col. 11, lines 43-49, that when the layout stage is implemented it is measuring the size and position of each frame (element)); wherein if the element is determined to have one or more children, then the element is determined to be a parent element, wherein if it is determined that the one or more children o f the parent element is to be measured, then only the parent element is stored in the measure tree (specification stage; frame tree), wherein executing the first procedure for measuring the element recursively executes the first procedure on one or more child elements of the parent element, wherein if it is determined that the one or more children of the parent element is not to be measured, then the one or more children of the parent element is determined to be an orphan, and wherein executing the first procedure for measuring the element does not recursively execute the first procedure on the orphan, and wherein the orphan is removed from the measure tree

(col.4, lines 49-62; col.6, lines 1-17 and 33-46; to recursively added elements to the frame tree wherein the frame tree is used in the specification stage to determining the size of elements within the tree; (the size of the elements (constraints) are measured by going through the tree in an order from leaf node to root, such that removal of consideration of an "orphan node" or leave node is performed in order to traverse the tree data structure, much like the same is traversal through a queue as currently claimed; col.11, lines 39-55; further layout (arrangement) of the elements is ordered by placing the root elements (parents) first then the children nodes (from root to leaf); col.12, lines 34-42); executing a second procedure for arranging the element , wherein the element is stored in the arrange queue (col. 2, lines 1-9; col. 4, lines 57-64; col. 11, lines 51-55, that the auto-layout engine arranges and repositions the frames (elements) as it traverses the tree to fill available space; col.4, lines 46-56; layout stage); and wherein the second procedure is invoked and executed independently from the first procedure (fig. 6, label 604; col. 11, lines 56-67 and col. 12, lines 1-13); computes a final size for the element, performs internal arrangement functions on the element if the element has no children an dif the element has children computes display positions for a child-element of the element, wherein the internal arrangement functions include font, alignment, and color operations affecting the appearance of the element and wherein the display positions comprise a coordinate of a shape representing the element (col.2, lines 20-27; col.9, lines 64-67; col.10, lines 1-24, 45-57; col.11, lines 1-8, 15-21 and 39-55; col.14, lines 27-36 and 41-55; col.15, lines 15-20; figures 5-8, item 712). (col.9, line 55 – col.11, line 41).

Breinberg does not specifically teach the use of queues, instead uses a different data structure a tree. It is defined in the specification of the immediate application in paragraph 83 that alternatively instead of a queue an array, heap, tree or other data structure may be used without departing from the scope. Since Breinberg teaches two different stages of execution the specification stage (where elements are sized “measured”; traversal through the tree data structure to measure graphical user interface elements, much like the same function as claimed but yet claiming to the use of a queue data structure) and the layout stage (where elements are positioned "arranged"; traversal opposite of the measuring through the tree data structure to be arranged with the autolayout engine, much like the same function as the claimed but yet claiming to the use of a queue data structure) and the use of trees and not the teaching of using queues instead of trees, Buxton is introduced to cure the deficiencies of Breinberg in such that Buxton teaches the use of another data structure; multiple queues for the purposes of generating customized graphical user interfaces for applications in an object –oriented environment; wherein the queues are used in the part of creating the graphical user interfaces (col.26, lines 31-67; col.27, lines 18-40 and 59-67 and col.28, lines 1-21). It would have been obvious to one of ordinary skill in the art at the time of the invention was made to combine Buxton into Breinberg, this is true because Buxton solves the problem of configuring graphical user interfaces by using data structures to automatically manage graphical elements, such as pertaining to the layout of elements to be presented to the user (col.2, lines 34-60). Breinberg solves this problem by using tree data structures (for storage of elements) to aid in the proper

layout of a graphical user interface elements for the creation of an interface while Buxton uses queue data structures (for storage of elements) to aid in the creation of graphical user interface wherein dynamically building and placing elements within. Both Breinberg and Buxton make use of a means to automatically layout elements of a graphical user interface.

As claim 2, Breinberg further teaches the first procedure returns a desired size for the element (fig. 6, label 606; col. 12, lines 14-23, it is inherent that after the calculation, the results to include the desired size will be returned).

As claim 3, Breinberg further teaches the first procedure computes desired sizes for child-elements of the element (fig. 6, label 606; col. 12, lines 14-23).

As claim 4, Breinberg further teaches the first procedure comprises determining whether a child-element requires computation of its desired size (col. 17, lines 14-22).

As claim 7, Breinberg further teaches signaling the element's need to be measured by the first procedure (fig. 4, label 404; col. 10, lines 18-24).

As claim 8, Breinberg further teaches the signaling step comprises calling a measure invalidation function (col. 2, lines 24-27).

As claim 9, Breinberg further teaches the signaling step further comprises setting a flag on the element (col. 13, lines 3-8).

As claim 10, Breinberg further teaches the signaling step comprises notifying the operating system (col. 13, lines 37-39).

As claim 12, Breinberg further teaches the element requests the measuring of all elements needing to be measured (fig. 4; label 404; col. 10, lines 18-24).

As claim 13, Breinberg further teaches signaling with a signal an element's need to be arranged by the second procedure (col. 2, lines 34-41, it is inherent that the size and position of the child frames depend on parent frame, therefor, when anyone of the child frames change a windows message is sent to arrange the child frames).

As claim 14, Breinberg further teaches the signal comprises calling an arrange invalidation function (col. 2, lines 24-27, it is inherent that a windows message will be Sent for all windows (elements) that need to be arranged).

As claim 15, Breinberg further teaches the signaling step further comprises setting a flag on the element (col. 13, lines 3-8).

As claim 16, Breinberg further teaches the element requests the arranging of all elements needing to be arranged (col. 2, lines 34-41, it is inherent that the size and position of the child frames depend on parent frame, therefore, when anyone of the child frames change a
windows message is sent to arrange all the child frames).

As claim 17, Breinberg teaches a computer storage medium having stored thereon a set of executable procedures callable by a computer application program for making ready for presentation a graphical element (col. 1, lines 59-62 and lines 64-67; col. 2, line 1), including at least: maintaining a measure queue storing a list of elements to be measured (col.4, lines 31-45 (specification stage) ; during the specification stage graphical user interface elements are specified according to parent and child relationships and their boundaries, also the size of the elements are specified

(col.6,lines 65-67; col.7,lines 1-16). The data structured used to access these elements is a tree data structure)); maintaining an arrange tree storing a list of elements to be arrange (col.4, lines 46-56 (layout stage; during the layout stage the method access a tree data structure to go through all of the graphical user interface elements using an autolayout engine that traverses the tree data structure; from here the graphical user interface is constructed with appropriate layout of elements (col.9, lines 37-47)); a first procedure for measuring the element (fig. 6, label 602; col. 2, lines 1-9; col. 11, lines 43-49, that when the layout stage is implemented it is measuring the size and position of each frame (element)); wherein if the element is determined to have one or more children, then the element is determined to be a parent element, wherein if it is determined that the one or more children o f the parent element is to be measured, then only the parent element is stored in the measure Queue (specification stage; frame tree), wherein executing the first procedure for measuring the element recursively executes the first procedure on one or more child elements of the parent element, wherein if it is determined that the one or more children of the parent element is not to be measured, then the one or more children of the parent element is determined to be an orphan, and wherein executing the first procedure for measuring the element does not recursively execute the first procedure on the orphan, and wherein the orphan is removed from the measure queue (col.4, lines 49-62; col.6, lines 1-17 and 33-46; to recursively added elements to the frame tree wherein the frame tree is used in the specification stage to determining the size of elements within the tree ; (the size of the elements (constraints) are measured by going through the tree in an order from leaf

node to root, such that removal of consideration of an "orphan node" or leave node is performed in order to traverse the tree data structure, much like the same is traversal through a queue as currently claimed; col.11, lines 39-55; further layout (arrangement) of the elements is ordered by placing the root elements (parents) first then the children nodes (from root to leaf); col.12, lines 34-42);a second procedure for arranging the element, wherein the element is stored in the arrange queue (col.4, lines 46-56; layout stage) wherein the second procedure at least determines whether the element has one or more children and performs internal arrangement functions on the element when the element has no children (col. 2, lines 1-9; col. 4, lines 57-64; col. 11, lines 51-55,.that the auto-layout engine arranges and repositions the frames (elements) as it traverses the tree to fill available space);

and wherein the first procedure and the second procedure are used to mange a layout of one or more graphical elements, and the second procedure is called and executed independently from the first procedure (fig. 6, label 604; col. 11, lines 56-67 and col. 12, lines 1-13); computes a final size for the element , performs internal arrangement functions on the element if the element has no children an dif the element has children computes display positions for a child-element of the element, wherein the internal arrangement functions include font, alignment, and color operations affecting the appearance of the element and wherein the display positions comprise a coordinate of a shape representing the element (col.2, lines 20-27; col.9, lines 64-67; col.10, lines 1-24, 45-57; col.11, lines 1-8, 15-21 and 39-55; col.14, lines 27-36 and 41-55; col.15, lines 15-20; figures 5-8, item 712). (col.9, line 55 – col.11, line 41).

Breinberg does not specifically teach the use of queues, instead uses a different data structure a tree. It is defined in the specification of the immediate application in paragraph 83 that alternatively instead of a queue an array, heap, tree or other data structure may be used without departing from the scope. Since Breinberg teaches two different stages of execution the specification stage (where elements are sized “measured”; traversal through the tree data structure to measure graphical user interface elements, much like the same function as claimed but yet claiming to the use of a queue data structure) and the layout stage (where elements are positioned "arranged"; traversal opposite of the measuring through the tree data structure to be arranged with the autolayout engine, much like the same function as the claimed but yet claiming to the use of a queue data structure) and the use of trees and not the teaching of using queues instead of trees, Buxton is introduced to cure the deficiencies of Breinberg in such that Buxton teaches the use of another data structure; multiple queues for the purposes of generating customized graphical user interfaces for applications in an object –oriented environment; wherein the queues are used in the part of creating the graphical user interfaces (col.26, lines 31-67; col.27, lines 18-40 and 59-67 and col.28, lines 1-21). It would have been obvious to one of ordinary skill in the art at the time of the invention was made to combine Buxton into Breinberg, this is true because Buxton solves the problem of configuring graphical user interfaces by using data structures to automatically manage graphical elements, such as pertaining to the layout of elements to be presented to the user (col.2, lines 34-60). Breinberg solves this problem by using tree data structures (for storage of elements) to aid in the proper

layout of a graphical user interface elements for the creation of an interface while Buxton uses queue data structures (for storage of elements) to aid in the creation of graphical user interface wherein dynamically building and placing elements within. Both Breinberg and Buxton make use of a means to automatically layout elements of a graphical user interface.

As claim 18, Breinberg further teaches the first procedure returns a desired size for the element (fig. 6, label 606; col. 12, lines 14-23, it is inherent that after the calculation, the results to include the desired size will be returned).

As claim 20, Breinberg further teaches at least a procedure for signaling the element's need to be measured (fig. 4, label 404; col. 10, lines 18-24).

As claim 21, Breinberg further teaches at least a procedure for signaling the element's need to be arranged (col. 2, lines 34-41, it is inherent that the size and position of the child frames depend on parent frame, therefore, when anyone of the child frames change a windows message is sent to arrange all the child frames).

As claim 23, Breinberg further teaches at least a procedure for requesting the measurement of all elements needing to be measured (fig. 4, label 404; col. 10, lines 18-24).

As claim 24, Breinberg further teaches at least a procedure for requesting the arrangement of all elements needing to be arranged (col. 2, lines 34-41, it is inherent that the size and position of the child frames depend on parent frame, therefore, when anyone of the child frames change a windows message is sent to arrange all the child frames).

As claim 26, Breinberg teaches a computer system for making ready for presentation a graphical element (fig. 3; col. 8, lines 31-34), the system comprising: a memory for storing executable program code; and a processor, functionally coupled to the memory, the processor being responsive to computer-executable instructions contained in the program code and operative to execute (col. 6, lines 1-7. The data about the frame (element) is contained in a data structure describing the position and dimensions of the specified frame (element)); maintaining a measure queue storing a list of elements to be measured (col.4, lines 31-45 (specification stage); during the specification stage graphical user interface elements are specified according to parent and child relationships and their boundaries, also the size of the elements are specified (col.6, lines 65-67; col.7, lines 1-16). The data structured used to access these elements is a

tree data structure)); maintaining an arrange tree storing a list of elements to be arrange (col.4, lines 46-56 (layout stage; during the layout stage the method access a tree data structure to go through all of the graphical user interface elements using an autolayout engine that traverses the tree data structure; from here the graphical user interface is constructed with appropriate layout of elements (col.9, lines 37-47));

a first executable procedure using the data structure for measuring the element, wherein the first executable procedure at least determines whether the element has one or more children determines, if the element has one or more children, whether the one or more children of the element is to be measured (col.4, lines 56-64);

and determines a size for the element based on the an element type for the element when the element has no children (fig. 6, label 602; col. 2, lines 1-9; col. 11, lines 43-49); wherein if the element is determined to have one or more children, then the element is determined to be a parent element, wherein if it is determined that the one or more children o f the parent element is to be measured, then only the parent element is stored in the measure Queue (specification stage; frame tree), wherein executing the first procedure for measuring the element recursively executes the first procedure on one or more child elements of the parent element, wherein if it is determined that the one or more children of the parent element is not to be measured, then the one or more children of the parent element is determined to be an orphan, and wherein executing the first procedure for measuring the element does not recursively execute the first procedure on the orphan, and wherein the orphan is removed from the measure queue (col.4, lines 49-62; col.6, lines 1-17 and 33-46; to recursively added elements to the

frame tree wherein the frame tree is used in the specification stage to determining the size of elements within the tree ; (the size of the elements (constraints) are measured by going through the tree in an order from leaf node to root, such that removal of consideration of an "orphan node" or leave node is performed in order to traverse the tree data structure, much like the same is traversal through a queue as currently claimed; col.11, lines 39-55; further layout (arrangement) of the elements is ordered by placing the root elements (parents) first then the children nodes (from root to leaf); col.12, lines 34-42). When the layout stage is implemented it is measuring the size and position of each frame (element)); and a second executable procedure using the data structure for arranging the element (col. 2, lines 1-9; col. 4, lines 57-64; col.-11, lines 51-551 that the auto-layout engine arranges and repositions the frames (elements) as it traverses the tree to fill available space); wherein the element is stored in the arrange queue (col.4, lines 46-56; layout stage); computes a final size for the element , performs internal arrangement functions on the element if the element has no children an dif the element has children computes display positions for a child-element of the element, wherein the internal arrangement functions include font, alignment, and color operations affecting the appearance of the element and wherein the display positions comprise a coordinate of a shape representing the element (col.2, lines 20-27; col.9, lines 64-67; col.10, lines 1-24, 45-57; col.11, lines 1-8, 15-21 and 39-55; col.14, lines 27-36 and 41-55; col.15, lines 15-20; figures 5-8, item 712). (col.9, line 55 – col.11, line 41).

Breinberg does not specifically teach the use of queues, instead uses a different data structure a tree. It is defined in the specification of the immediate application in

paragraph 83 that alternatively instead of a queue an array, heap, tree or other data structure may be used without departing from the scope. Since Breinberg teaches two different stages of execution the specification stage (where elements are sized “measured”; traversal through the tree data structure to measure graphical user interface elements, much like the same function as claimed but yet claiming to the use of a queue data structure) and the layout stage (where elements are positioned "arranged"; traversal opposite of the measuring through the tree data structure to be arranged with the auto layout engine, much like the same function as the claimed but yet claiming to the use of a queue data structure) and the use of trees and not the teaching of using queues instead of trees, Buxton is introduced to cure the deficiencies of Breinberg in such that Buxton teaches the use of another data structure; multiple queues for the purposes of generating customized graphical user interfaces for applications in an object –oriented environment; wherein the queues are used in the part of creating the graphical user interfaces (col.26, lines 31-67; col.27, lines 18-40 and 59-67 and col.28, lines 1-21). It would have been obvious to one of ordinary skill in the art at the time of the invention was made to combine Buxton into Breinberg, this is true because Buxton solves the problem of configuring graphical user interfaces by using data structures to automatically manage graphical elements, such as pertaining to the layout of elements to be presented to the user (col.2, lines 34-60). Breinberg solves this problem by using tree data structures (for storage of elements) to aid in the proper layout of a graphical user interface elements for the creation of an interface while Buxton uses queue data structures (for storage of elements) to aid in the creation of

graphical user interface wherein dynamically building and placing elements within. Both Breinberg and Buxton make use of a means to automatically layout elements of a graphical user interface.

As claim 27**,** Breinberg further teaches the data structure comprises: a first value representing the desired size of the element (col. 2, lines 26-27; col. 14, lines 52-55, it is inherent that the attributes is the value for the size); a second value representing the computed size of the element (col. 2, lines 26-27; col. 14, lines52-55, it is inherent that after the result of the method/function call, the returned value is the computed size value for the element); a first flag for triggering measurement of the element (col. 10, lines 3-20); and a second flag for triggering arrangement of the element (col. 10, lines 45-57).

As claim 28, Breinberg further teaches the first executable procedure returns a desired size for the element (fig. 6, label 606; col. 12, lines 14-23, it is inherent that after the calculation, the results to include the desired size will be retuned).

As claim 29, Breinberg further teaches the first executable procedure computes desired sizes of child-elements of the element (fig. 6, label 606; col. 12, lines 14-23).

As claim 32, Breinberg further teaches using the first flag for signaling the element's need to be measured by the first executable procedure (fig. 4, label 404; col. 10, lines 18-24).

As claim 33, Breinberg further teaches using the second flag for signaling the element's need to be arranged by the second executable procedure (col. 2, lines 34-41, it is inherent that the size and position of the child frames depend on parent frame. Therefore, when anyone of the child frames change a windows message is sent to arrange all child frames).

5. Claims 11 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Breinberg in view of Buxton in further view of Lupu (US Pub 2004/0100480).

As claim 11, Breinberg as modified by Dando does not teach the signaling step comprises notifying the element's parent-element. However, Lupu teaches the signaling step comprises notifying the element's parent-element (par [0007]). Therefore, it would have been obvious to one ordinary skill in the art the time the invention to modify Breinberg by having signaling step to notify the element's parent-element as taught by

Lupu in order to provide constant communication between window objects (elements) enhancing the over all functionality.

As claim 22, Breinberg as modified by Dando does not teach the procedure for signaling to a parent element the child element's need to be measured. However, Lupu teaches the procedure for signaling to a parent element the child element's need to be measured (par [0007]). Therefore, it would have been obvious to one ordinary skill in the art the time the invention to modify Breinberg by having the procedure for signaling to a parent element the child element's need to be measured as taught by Lupu in order to provide a functional interface between modules utilizing window messages constantly updating the status of each window object (element).

(Note :) **It is noted that any citation to specific, pages, columns, lines, or figures in the prior art references and any interpretation of the references should not be considered to be limiting in any way. A reference is relevant for all it contains and may be relied upon for all that it would have reasonably suggested to one having ordinary skill in the art. In re Heck, 699 F.2d 1331, 1332-33, 216 USPQ 1038, 1039 (Fed. Cir. 1983) (quoting In re Lemelson, 397 F.2d 1006,1009, 158 USPQ 275, 277 (CCPA 1968)).**

Response to Arguments

Applicant's arguments with respect to claims 1-4, 7-18, 20-24, 26-29 and 32-33 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Inquires

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Nicholas Augustine whose telephone number is 571-270-1056 and fax is 571-270-2056. The examiner can normally be reached on Monday - Friday: 9:30am- 5:00pm Eastern.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Weilun Lo can be reached on 571-272-4847. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Steven B Theriault/
Primary Examiner, Art Unit 2179

/Nicholas Augustine/
Examiner
Art Unit 2179
May 18, 2009